Presenting:

Hive – A Petabyte Scale Data Warehouse Using Hadoop

By: *Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy*

And

A Comparison of Approaches to Large-Scale Data Analysis

By: *Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samule Madden and Michael Stonebraker*

In

A Big Data Paper

By: Tom Gonzalez

May 11, 2015

Main Idea of Hive

Hive is an open-source data warehousing solution built on top of Hadoop meant to address certain intrinsic issues of Hadoop. Currently Hive is still a work in progress that is actively worked on by Facebook as well as several other contributors. The Hive system addresses the need for a flexible data infrastructure that can handle large volumes of data and cater to diverse applications/users in a scalable, cost-efficient manner. Hadoop is an open source project that provides infrastructure for big data using a MapReduce computing model.

Hadoop can be difficult to use since end users of Hadoop have to write map-reduce programs for simple tasks, unlike SQL. In order to facilitate the analysis of their data, Facebook developed Hive. Hive was created with the vision of improving the query capabilities of Hadoop by bringing the table, column, and partition concepts of SQL to the unstructured world of Hadoop without sacrificing the extensibility and flexibility of Hadoop. As a result, Hive supports queries expressed in a SQL-like declarative language (HiveQL) which are complied into MapReduce jobs that are then executed by Hadoop.

How is Hive Implemented

As previously mentioned HiveQL, is very similar to SQL and thus can easily be interpreted by anyone already familiar with SQL. Like SQL, Hive structures data into tables columns rows and partitions. Each table has a certain number of rows, each row a certain number of columns, and each column has an associated type. Hive supports all the major primitive data types such as integers, floats, doubles and strings as well as complex data types such as maps, lists and structs. Complex data types can be nested arbitrarily to construct types that are even more complex.

For tables created within Hive there is a default serializer and deserializer but for data prepared in other programs or that may be legacy data Hive provides a jar system. This allows the incorporation of outside data without needing to transform the data. Hive also uses a Bucket storage unit concept which is a file within the leaf level directory of a table or partition.

My Analysis of Hive and its Implementation

Hive gives structure to unstructured data and allows a greater degree of user-friendliness. Considering how Hive manages the combination of familiar SQL techniques and structure with big data, one can immediately recognize the system's potential. I, as an individual familiar with SQL, like this aspect of Hive and would be interested in watching its application first-hand. Certain limitations of Hive have been expressed such as how only equality predicates in join predicates are supported and that the joins have to be specified using the ANSI join system. Also, the method in which inserts are done poses another limitation as Hive does not currently support inserting into an existing table or data partition. Instead all inserts overwrite the existing data. According to paper on Hive (cited at the beginning) these restrictions have not been a problem though, as a case has rarely been seen where a query cannot be expressed as an equi-join.

Main Ideas of Comparison Paper

In *A Comparison of Approaches to Large-Scale Data Analysis* the MapReduce system is compared and evaluated against a parallel SQL database management system. Each system entails "cluster computing" which is using large numbers of processors in parallel to solve a computing problem. In the aforementioned paper, the performance of each system is analyzed against a benchmark which is comprised of a series of tasks. The results of this experiment have shown that it takes longer to load data into and tune the execution of parallel database management systems than it does for a MapReduce system. However, a parallel database management system performs significantly better than a MapReduce system.

How are the Ideas of comparison paper implemented

To compare both systems, the performance of each is measured over a cluster of 100 nodes. Specifically the Hadoop system was compared against Vertica and a second parallel SQL DBMS from a major relational vendor. For the Parallel DBMSs, data is required to fit the relation model of rows in columns while the MapReduce model does not require any relational schema, the MR programmer is free to structure their data in any manner (or even without any manner). All DBMSs use indexes to accelerate access to data the MR frameworks, in contrast, do not provide any kind of built-in indexes. For a MR framework the programmer must implement any desired indexes to accelerate access to data.

Analysis of Comparison Paper Ideas/Implementation

The most important difference between parallel DBMSs and the MR system would appear to be that the MR has a "commitment to a schema later or schema never paradigm.". This "commitment" has a number of related consequences that result in the system being out-performed by a parallel DBMS. Considering that the MR system of Hadoop is fairly new and that there are platforms such as Pig and Hive being constructed on top of it to address some of these shortcomings, I would credit the system as still being in its infancy while the parallel DBMSs have had the advantage of decades of development. I think that with further development the MR system will surpass the parallel DBMSs as the quantity of data under analysis continues to grow (and the cost of powerful hardware continues to decline).

Comparison between Hive/Other Paper

Hive embodies the convergence between the MR system and parallel DBMSs discussed at the end of the comparison paper. Hive takes a SQL-added approach to structuring big data that combines familiar concepts with unprecedented quantities of data. In Facebook's case the use of Hadoop enabled a reduction in the time taken to perform some daily data processing over the use of a commercial RDBMS. Facebook had a need for a certain degree of flexibility which a RDBMS could not offer.

Main Ideas of Stonebraker talk

In 2005 a team led by Stonebraker published a paper One Size Fits All – An Idea Whose Time Has Come and Gone (2005 paper). From 1970-2000 RDBMS were thought of as the answer to any question but in 2005, Stonebraker and his Team quickly realized that the RDBMS could not fit all. At this point, in 2015, one size fits none, typical relational databases are now obsolete. There is different SQL implementation in OLTP/Data Warehouse market taking place. There is also a NoSQL market, with 100+/- vendors that have no standards and a "potpourri of data models and architectures". For complex data analytics, business analysts use data warehouses with things like Cognos to perform certain analytic but within ten years data scientists will replace business analysts. Column or array stores are predicted to get this analytics market. As it currently stands there is a huge diversity of database engines, all oriented towards specific application, of which row stores are good for none. Furthermore, new hardware will result in great deal of new database implementation as the bottleneck of databases, networking, is overcome.

Advantages/disadvantages of Hive in context of comparison paper and Stonebraker talk.

Hive, as said before, seems to be one of the up-and-coming platforms. It appears as a product tailored for Facebook at the moment (but I'm sure it has many other applications, and that this number of applications will only grow as time progresses). Hive takes the old concepts and implements them in a new system which when refined should allow for more efficient analytics. Unfortunately, while Hive will improve as time goes on, so too will the competition and other products may surpass Hive's capabilities, which would then render Hive obsolete.